Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Information Flow Based Decomposition of Decision-Making Problems Involving Partial Observability

Rakshita Agrawal *, Jay H. Lee *, Matthew J. Realff

* *Department of Chemical and Biomolecular Engineering, Georgia Institute of Technology,*

Atlanta, GA 30332-0100, USA

Abstract: Industrial planning and scheduling decisions are often inter-dependent. For example, planning level capacity allocation decisions affect production scheduling. While independent decision rules fail to address the above mentioned inter-dependence, simultaneous consideration of all interactions leads to a very large problem, which is oftentimes computationally intractable. In this study, we demonstrate, in the context of a machine maintenance problem for a reentrant flow system, a middle-ground approach by recognizing paths of strong information flow and then systematically decomposing the problem to be able to obtain a computationally tractable problem yielding a near-optimal decision policy. In the process, we make combined use of rigorous probability theories, approximate dynamic programming and simulation based rules.

1. INTRODUCTION

Decision making in manufacturing takes place at multiple levels which differ in terms of time scales. High level planning decisions like capacity allocation, resource acquisition and maintenance, directly affect the lower level scheduling decisions like job release and inspection. Scheduling decisions also affect planning decisions by providing information on equipment health and capacity requirements. Therefore, the decisions at two levels are inter-dependent. Furthermore, these decisions are to be taken in the presence of uncertainty from various operational and external sources and based on limited information. Despite the obvious inter-dependence among such decisions, they are mostly treated independent of one another, in the existing literature.

One example of such complex manufacturing system is a semi-conductor wafer manufacturing fab where delays due to testing and shutdowns translate directly into huge loss in revenue. Besides the abovementioned features, wafer manufacturing comes with the added complexity of re-entrant flow lines, which accentuates the inter-dependence between machine maintenance and job release scheduling.

General machine maintenance problems have been formulated and solved as Markov Decision Processes (*MDP*). In case of preventive maintenance, the states are not fully observable. This problem is discussed in (Smallwood and Sondik, 1973). Past researches in wafer-fabrication scheduling are mostly aimed at global job-shop scheduling. Gupta and Shivakumar (2006) provide a comprehensive overview of scheduling techniques in a semiconductor manufacturing process. Attempts have also been made to capture, specifically, the re-entrant flow structure of the problem. Shen and Leachman (2003) have proposed a stochastic dynamic programming model for scheduling new releases. They captured the re-entrant flow structure characteristic of wafer manufacturing by a stochastic linear quadratic (SLQ) model. To address the fact that, not every intermediate can be tested during manufacturing, many statistical control studies have been conducted for optimal sampling policies. One such work is by Nurani et al. (1994). They have suggested ways to develop an optimal sampling strategy for defect inspection in semi-conductor wafers using real-life data from different fabs. Their work gives useful insights into modeling of process drifts and defect-yield relationship. With the rich literature available on independent studies of these decisions, we adopt the objective of studying how these decisions affect one another. By means of a single machine problem with re-entrant flow, we analyze whether and by how much, combining the decisions would improve the overall performance. The rigorous application of formal methods such as stochastic dynamic programming soon turns intractable for practical size problems. This calls for a systematic decomposition of the combined problem.

The article is organized as follows. Section 2 contains a detailed description of the system. Section 3 presents the mathematical formulation and the computational complexity of the resulting problem. In section 4, we address the issue of partial observability and introduce the formal representation of the same. Section 5 contains a description of various solution methods used and the discussion on results. In this section we consider some industrial policies, followed by a rigorous treatment of the problem and then introduce the decomposition scheme. The conclusions are summarized in section 6.

2. PROBLEM DESCRIPTION

2.1 System details

A re-entrant flow shop is characterized by a job going through the same operation more than once. Thus, jobs at various stages of processing compete for the same resources. For the purpose of illustration, we use a hypothetical machine capable of depositing identical layers, one layer at a time (see Fig. 1). It operates on a single wafer at a time. The product of interest is a wafer with 3 identical layers deposited. The wafers are fed one at a time. While one wafer is being operated on, the others are waiting at the entry point called 'queue'. The queue at any time can contain 3 types of intermediate/unprocessed jobs
a_0 = bare wafer or unprocessed job
a_1 = wafer with 1 layer
a_2 = wafer with 2 layers

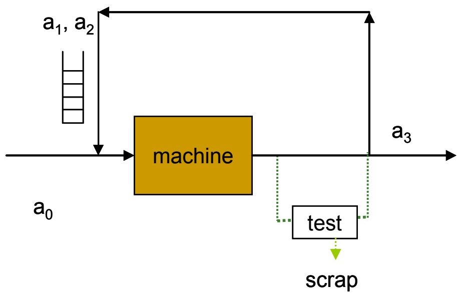

Fig. 1. Schematic representation of a single work-station facilitating re-entrant flow

The cost of machine operation is substantial and hence, a_2 is a more expensive intermediate than a_1, which is more expensive than a_0. We assume that there exists unlimited raw wafer supply and all final products are tested due to quality control reasons. We also assume that negligible time is needed to test intermediates and perform machine maintenance, for which some costs are incurred.

2.2 Process drift

The machine is prone to degradation in time and hence produces bad layers once in a while. The defect generation is random in nature and the machine in good working condition may also produce defects but at a much lower frequency as compared to when it is in a degraded state. In essence, the good machine state differs from bad ones in terms of rates of defect generation.

In their statistical studies, Nurani et al. (1994) show that the number of defects per unit time fluctuates about a constant mean at the beginning and then keeps increasing with time. The different states of the machine health are not directly observable and the only information that rests with the operator is whether a defect has occurred or not. In this study, to model the time dependency of the defect rate, the machine performance is approximated with a Markov switching model containing 3 regimes of machine health as in Figure 2(a). The transition probabilities of the Markov Chain are as shown in Figure 2(b) and these switching probabilities are considered throughout this study. The 3 regimes differ in terms of their defect rate. Regime 1 is the best possible machine state with lowest defect rate, while regime 3 is the worst state. It is also the absorbing state, i.e., the system remains at regime 3 until a maintenance job is performed to bring it back to the best state.

For the 3 layer product of our interest, we assume that a defect in any layer renders the entire product defective. No distinction is made between a defect in the 1st layer and a defect in the 3rd layer in this study. Since the cost of maintenance is high, defect generation is inevitable.

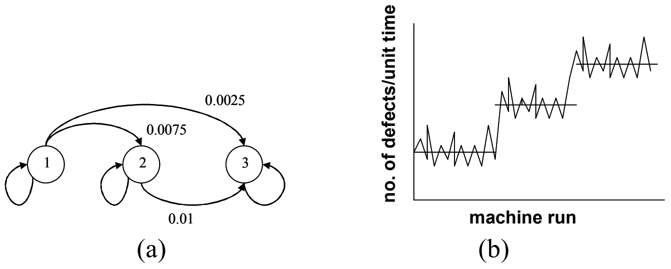

Fig. 2 (a) Underlying Markov Chain at the work-station; (b)-regime machine model with Markov switching

Intuitively, it should be best to run expensive jobs (a_2) when the machine is close to regime 1 (with a lower chance of making a defect). Cheaper intermediates (a_0 and a_1) should be processed when the defect rate is high. To keep defects from propagating, these cheaper intermediates must be tested and reworked/scrapped when found defective. Keeping this intuition in mind, we present the mathematical formulation of the problem in the following section.

3. PROBLEM FORMULATION

<u>System State</u>
The system at any time is fully characterized by the following
$x = [n_1\ n_2\ d_1\ d_2\ regime]$

n_1 - number of a_1 in the queue
n_2 - number of a_2 in the queue
d_1 - number of defective a_1's in the queue
d_2 - number of defective a_2's in the queue
regime – integer representing the regime of machine health

The state space would consist of all possible combinations of the above parameters. For instance, if the queue length is limited to 5 for n_1 and 5 for n_2, then n_1 can have 6 possible values (0, 1, 2..,5). Similarly, n_2 can have 6 possible values. For a particular value of n_1 say 3, d_1 can hold 4 possible values from 0,1,2,3. For the given queue lengths then, the number of possible combinations for $[n_1\ n_2\ d_1\ d_2]$ is 441. With 3 regimes for the machine, the size of state space is 1323 (441 x 3). If there is a common queue with length 10, the size of the state space becomes 3003 states (1001 x 3). In all future analyses we have considered a combined queue length of 10 unless otherwise mentioned.

Action/ decisions
$u = [schedule\ \ test\ \ maintain]$
schedule - job scheduling decision (admit a_0 ,a_1 or a_2)
test – binary, test (1) the processed wafer or not (0)
maintain – binary, run a maintenance job on the machine(1) or not (0)
Assuming all final products are tested
Size of action space = ((3 x 2)-1) x 2 = 10

Uncertainty
a). Machine regime switching - As shown in Figure 3, the machine can switch between regimes with certain probabilities in a non-deterministic manner.

b). Defect generation - Defect generation is probabilistic and the defect probability is set by the regime in which the machine is operating.

c). Error propagation- Since not all intermediates are tested, the queue can contain defective intermediates, designated as d_1 and d_2 in the state description. Probability that a defective intermediate is picked and operated upon is given by q:

$q = \frac{d_1}{n_1}$ For a_1 being operated

$q = \frac{d_2}{n_2}$ For a_2 being operated

Objective
A profit measure is maximized. In this study we look to maximize infinite horizon discounted profit/ reward:

$$\max_{I_{mt}, I_{Tt}, I_{a_0t}, I_{a_2t}} \sum_{t=1}^{\infty} \gamma^t (C_p I_{pt} - C_{oi} - C_m I_{mt} - C_T I_{Tt} - C_{a_0} I_{a_0t})$$
(1)

where,

γ = discounting factor	(0.99)
C_p = product price	(1000)
C_m = cost of maintenance	($10C_p$)
C_T = cost of test	($0.1C_p$)
C_{a0} = cost of raw material	($0.1C_p$)
C_{o1} = cost of processing a_0	($0.05C_p$)
C_{o2} = cost of processing a_1	($0.1C_p$)
C_{o3} = cost of processing a_2	($0.2C_p$)

The values indicated in the parenthesis are parameter values for all simulation purposes throughout this work unless otherwise mentioned. These values are reasonably chosen to represent the trade-offs between different cost heads in a typical manufacturing environment. Since the queue length is constrained, holding cost/ WIP cost is not considered.

All I's (I_{pt}, I_{mt}, I_{Tt}, I_{a0t}) are binary and are equal to 1 when a non-defective product is produced, when a maintenance job is run, when a finished job is tested and when a_0 is run at time *t,* respectively. C_{oi} is the processing cost when a_i is processed.
Different values of C_{oi} are considered for various policies to analyze the associated tradeoffs. Since scheduling, testing, and maintenance are part of the combined decision making, they should be chosen so that the overall profit is maximized.

The abovementioned problem falls under the vast domain of Markov Decision Processes and can be solved as a discounted infinite horizon problem using value iteration proposed by Bellman (Bertsekas, 1995), for reasonable size problems. Problem size is mostly governed by the queue length. The *Bellman equation* is as shown below,

$$V(s) = \max_{a \in A} \{ r(s,a) + \gamma \sum_{s' \in S} p(s' | s,a) V(s') \} \qquad (2)$$

where, *V(s)* is the optimal value function for starting at state *s, r(s,a)* is the reward when action *a* is taken in state *s* and *p(s'|s,a)* is the transition probability from *s* to *s'*. For finite queue sizes, the optimal value function would give an optimal stationary policy for each state. However, the use of such a policy requires that the states be fully known to the decision- maker, which is not realistic. For this reason, we refer to the policy that results from solving (2) as Fully Observable Markov Decision Process (*FOMDP*) policy. The performance of the *FOMDP* policy for the queue length of 10 derived via value iteration is reported in figure 3. However, the challenge arises from the fact that a part of the state is not exactly known to the decision-maker. This is discussed in the following section.

4. THE PARTIAL OBSERVABILITY

As mentioned earlier the decision-maker does not see the regime that the machine is in, at any time other than when the machine is just serviced. Also, a part of the state is not known to the decision-maker since d_1 and d_2 are the undetected defects accumulated in the system. To sum up, the elements that are directly observed are n_1, n_2 and defect status of each tested product (1 if a defect occurred, 0 otherwise).

Due to the mentioned partial observability, the *FOMDP* solution is not applicable. The lack of full observability in MDPs is handled rigorously by the formulation of Partially Observable Markov Decision Processes (*POMDPs*), which are briefly described in the following section.

4.1 POMDP description

A partially observable Markov decision process (*POMDP*) describes a stochastic control process with partially observable (hidden) states. Formally, it corresponds to a tuple (*S, A, Θ, T, O, R, Π*):

- *S* – set of Markov states / state space
- *A* – set of actions / action space
- *Θ* – set of observations / observation space
- *T – p(s'|s,a)* transition probability
  Probability of being in *s'* at *t+1*, when action a is taken from state *s* at time *t*
- *O- p(o|s',a)* observation probability
  Probability of getting observation o at time *t+1*, when action a is taken at time *t* and state s' is reached at time *t+1*
- *R – r(s,a)* Reward when action a is taken in state s at time *t*
- *Π – p(s)* Initial probability distribution at *t=0*

Concept of Information state MDP

The *POMDP* problem shown above can be represented as an equivalent information state *MDP* where the information state contains

- a prior belief b_0 on states S at time 0
- a complete history of actions and observations starting from time 0

Due to the Markov property, it turns out that a belief state *b(s)*, i.e., the conditional probability of being in state *s* at time *t*, is a sufficient information state for our problem. The conditions for a sufficient information state can be found in (Hauskrecht, 2000). Therefore, the Bellman equation (2) of section 3.1 becomes

$$V(b) = \max_{a \in A} \{ \sum_{s \in S} r(s,a) b(s) + \gamma \sum_{o \in \theta} \sum_{s \in S} p(o \mid s,a) b(s) V(b') \} \qquad (3)$$

Belief-states can be updated via the following recursive equation obtained by applying Baye's rule:

$$b'(s') = \frac{p(o \mid s',a) \sum_{s \in S} p(s' \mid s,a) b(s)}{p(o \mid b,a)} \qquad (4)$$

The partial observability, thus converts the original problem into a continuous state Fully Observable *MDP* (*FOMDP*), where the state dimension is one less than the size of the original state space. The belief-states are continuous since they contain the probability values, which are continuous numbers between 0 and 1. In the following section we extend the problem formulation to accommodate the partial observability.

4.2 Formulation

The state space, action space, objective and the reward vector are the same as that in section 3.1 where the fully observable problem was introduced. The *POMDP* problem definition is continued below:

- Observation space Θ

[n_1 n_2 defect (if tested)]
For $n_1 + n_2 <= 10$, and three values for defect (0,1,2)
where, 0 - no defect occurred
1 - defect occurred
2 – no information
and size of the observation space is 198

- Observation probability matrix *p(o|s,a)*

p(o|s,a) is a matrix that maps a state-action pair to the probabilities of various possible observation comes. For example, for *s* = [1 0 1 0 1], possible observations are [1 0 *defect*], [1 0 *no_defect*] and [1 0 *no_test*].

- State transition probability matrix *p(s'|s,a)*

For a queue length of 10, *p(s'|s,a)* is a 3003 x 3003 matrix incorporating the 3 sources of uncertainty mentioned in section 3. There would be ten matrices corresponding to ten actions.

- Stage- wise reward *R(s,a)*

Includes product price, processing cost, raw material cost, inspection cost, maintenance cost and holding cost

Computational complexity

As seen above, the continuous state *MDP* for the problem described in section 3.1 would have a state dimension of 3003 with 10 actions. A problem of this magnitude is difficult to solve using exact POMDP solution methods. Furthermore, the problem size would increase exponentially with increase in queue length and so will the complexity. Presented below are some intuition based practical policies which serve as a good starting point for the analysis of the above problem.

5. SOLUTION METHODS

5.1 Solution using intuition based heuristics

5.1.1 Maintenance and inspection policies

- Periodic maintenance and inspection

The simplest industrial policy is to run machine maintenance every *m* runs and run a job inspection after every *t* jobs/machine runs, where *m* and *t* are parameters. This policy is completely oblivious to the information about machine health that the job inspection provides. Therefore a better policy based on the obtained information is as follows.

- Information based maintenance and inspection

Since the defect probabilities change with the machine regime, results of testing intermediates or finished products gives some information about the machine health. Therefore, at any time we can maintain conditional probability distribution over machine regimes and updating it using Baye's rule, whenever a new observation is made. If the 'maintenance and inspection only' problem is defined as below

State *s* – integer, machine regime
$S = \{1,2,3\}$
Action *a* – binary; maintain (1), do not maintain (0)
$A = \{0, 1\}$
Observation – binary; defect (1) , no defect (0)
$\Theta = \{0,1\}$
then the probability distribution over the machine regimes ($p(s)$) is updated as below:

$$p(s' \mid s,0) = \begin{bmatrix} 0.99 & 0.0075 & 0.0025 \\ 0 & 0.99 & 0.01 \\ 0 & 0 & 1 \end{bmatrix} \qquad p(s' \mid s,1) = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

$$p(o \mid s',a) = \begin{bmatrix} 0.05 & 0.1 & 0.5 \end{bmatrix}$$

$$p(s') = \sum_{s \in S} p(s) p(s' \mid s,a) \qquad \textit{if not inspected}$$

Use equation 4 if inspected.

Then, one candidate for information based maintenance and inspection policy can be stated as:

$if\ p(1) < p\_m$ *maintain*
$if\ max_s(p) < p\_t$ *inspect*

The above emphasizes the fact that once the knowledge of machine regime falls below a certain limit (p_t), we should run an inspection and if the probability of being in the best regime goes below p_m, maintenance should be performed.

5.1.2 Job release policies

- Sequential feeding

In order to avoid building up intermediates in the system and reaping early rewards, a greedy policy would be to have only one job in the system and process it until completion.

- Information based job release scheduling

Due to the difference in the values and processing costs of the intermediates, a better policy would recommend running a_2 when the machine is just serviced followed by a_1 and a_0. The policy is summarized below:

$if\ t_2 < p(1) \le t_3$ *run* a_2
$if\ t_1 < p(1) \le t_2$ *run* a_1
else *run* a_0

where t_1, t_2 and t_3 are parameters. The queue length constraint is resolved by picking the next best option.

We apply this heuristic policy to the following three cases having different defect probability values corresponding to the three machine regimes show in Table 1.

Case	**Defect probabilities** **[$p(1\|1)$ $p(1\|2)$ $p(1\|3)$]**
1	[0.05 0.1 0.5]
2	[0.05 0.25 0.75]
3	[0.05 0.5 1]

Table 1. Different cases pertaining to different defect probabilities in three machine regimes

The performance of some of the above policies is shown in figure 3 and the best parameter values for the policies are reported in table 2. The profit values are an average over 200 experiments using the optimal policy or policy corresponding to best parameter values. The computations were carried out in MATLAB (version R2007a) environment on a 3.00 GHz PC.

As is seen from the results, different cost and model parameters lead to different trade-offs in view of the decisions in question. The most notable result is that the difference in processing costs has higher impact on overall performance than the defect probabilities (as seen in figure 3(b)). The first series corresponds to the *FOMDP* solution. It is the theoretical upper bound, which cannot be achieved. The periodic policy generally performs very poorly stressing the need for using the information for decision-making. The improvement made by using the information increases with increasing difference in the defect probabilities of the machine regimes, i.e. from cases 1 through 3. This is because of the fact that the more different the regimes are the better the conditional probability estimates.

The best parameter values generally show a trend toward increased service runs and inspection with increasing defect probabilities. This number reduces with increasing processing costs, which is intuitive since the profit margins are lower to make up for the maintenance and inspections costs. A few anomalies are attributed to the stochastic outliers. For most of the policies, a sequential feeding schedule is optimal.

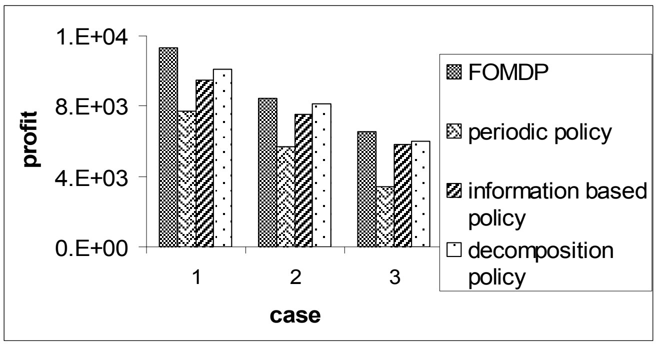

Fig. 3(a)

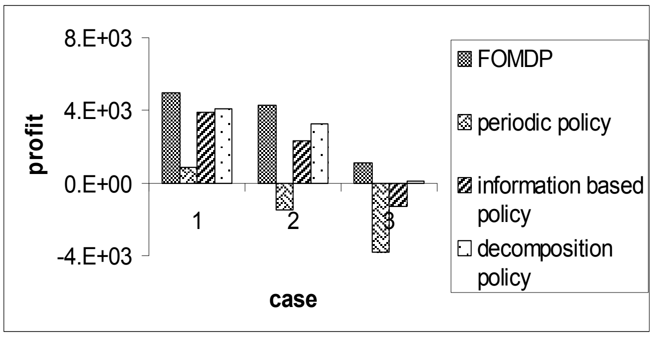

Fig. 3(b)

(a) Comparison of performances of different policies for three sets of processing costs as mentioned in table 1 and processing costs [C_{o1} C_{o2} C_{o3}] = [0.01 0.05 0.1]C_p
(b) For processing costs [C_{o1} C_{o2} C_{o3}] = [0.05 0.1 0.2]C_p

5.2 Solution using a decomposition scheme

Given the large computational complexity, we propose to decompose the problem by exploiting the basic problem structure. The machine maintenance decision is affected by testing, since the results of testing give information about machine regime, which is not directly observable. Also, testing and maintenance decisions directly affect job scheduling, while the reverse is not true. Job scheduling does not have a direct impact on machine maintenance in this particular problem setup. In addition, test decisions are only weakly dependent on the job being processed, the dependence coming from the need for picking out defective intermediates. Therefore, we aim at optimizing the maintenance and test decisions independent of job scheduling decision. The optimal policy thus obtained may then be used for good job release schedules and superimposing additional inspection.

The *POMDP* sub-problem involving the maintenance and inspection decisions, which is formally presented in section 5.1.1, is solved using value iteration in (Agrawal et al, 2007) . This sub-problem does not contain the re-entrant structure of the problem. Therefore, every non-defective layer reaps a reward, while in the overall problem a_3 reaps a reward C_p if and only if all three layers are defect free. Therefore, we need the real value of one non-defective layer to solve the sub-problem. It is obvious that the value will be less than $0.33C_p$, since good layers are wasted on defective products. To be able to calculate the average value of a good layer, we use the information based policy and calculated the average value as below:

$$avg.value\,of\,a\,good\,layer = \frac{total\,number\,of\,non-defective\,products}{total\,number\,of\,non-defective\,layers} C_p \quad (9)$$

The average value for various cases was between 267.8 and 294.7. The average value is taken to be 280 for the decomposition scheme. After obtaining a sub-optimal maintenance and test schedule independent of job scheduling, a heuristic rule can be employed for the job scheduling and additional inspection. As mentioned earlier in section 2.1, the expensive intermediates must be processed, when the machine is close to a good regime. Therefore, the information based job release schedule and inspection (section 5.1.2) is used. The results are reported in figure 3. The decomposition performs well in most scenarios. The performance gap increases with the increase in the value of information, i.e., the better quality of information.

Case	**Periodic maintenance and inspection**		**Information based maintenance and inspection**	
	Service every *m* runs	Inspect every *n* runs	Service if $p(1) < p\_m$	Inspect if $\max(p) < p\_t$
$[C_{o1}\ C_{o2}\ C_{o3}] = [0.01\ 0.05\ 0.1] * C_p$				
1	1050	300	0.2	0.4
2	600	1050	0.5	0.5
3	500	400	0.7	0.6
$[C_{o1}\ C_{o2}\ C_{o3}] = [0.05\ 0.1\ 0.2] * C_p$				
1	250	200	0.5	0.6
2	500	800	0.3	0.5
3	1000	250	0.5	0.9

Table 2. Best parameter values for the heuristics

5.3 Rigorous Solution of the POMDP for the Coupled Problem usingPERSEUS

PERSEUS (Spaan & Vlassis, 2005) is a randomized point based value iteration algorithm, which falls under the umbrella of point based methods for solving POMDPs. The key concept is that the value function for discounted infinite horizon POMDPs can be approximated well with a piecewise linear and convex function (Sondik ,1978). The details are omitted due to the space limitation.

We applied PERSEUS to the fully coupled inspection / maintenance / scheduling problem. The performance of PERSEUS using sample belief sizes |B| of 2500 belief points and 10,000 belief points are shown in figure 4. The problem was solved for cases 2 and 3 in plot 3(b). The results show that the decomposition scheme compares well with the results obtained using the rigorous method. It must be noted that improvement can be made in PERSEUS using bigger |B| and smaller convergence tolerances. The results reported in figure 4 are those obtained for reasonable computational expense.

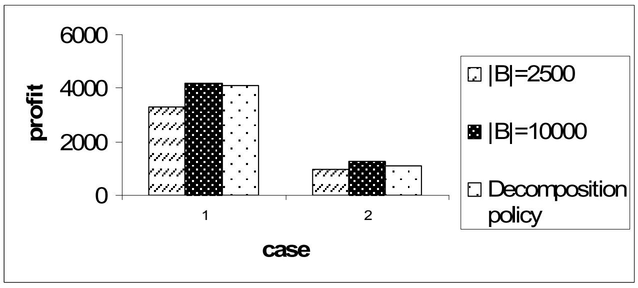

Fig. 4. Comparison with performance using PERSEUS

REFERENCES

Agrawal, R, J. H. Lee and M. J. Realff (2007). “Complex decision making at a re-entrant flow station under uncertainty.” *8th International IFAC symposium on dynamics and control of process systems* **3:** 210- 215

Bertsekas, D. P. (1995). *Dynamic Programming and Optimal Control*, Athena Scientific, Belmont, Massachusetts

Gupta, A. K. and A. I. Sivakumar (2006). "Job shop scheduling techniques in semiconductor manufacturing." *The International Journal of Advanced Manufacturing Technology* **27**(11-12).

Hauskretch, M. (2000). "Value-function approximations for partially observable Markov decision processes." *Journal of Artificial Intelligence Research* **13**: 33-94.

Nurani, R. K., R. Akella, A.J. Strojwas, R. Wallace, M.G. McIntyre, J. Shields and I. Emami (1994). " Development Of An Optimal Sampling Strategy For Wafer Inspection." *Extended Abstracts of 1994 International Symposium on Semiconductor Manufacturing.*

Shen, Y. and R. C. Leachman (2003). "Stochastic Wafer Fabrication Scheduling." *IEEE transactions on semiconductor manufacturing* **16**(1): 2-14.

Smallwood, R. D. and E. J. Sondik (1973). "The Optimal Control of Partially Observable Markov Processes over a Finite Horizon." *Operations Research* **21**(5): 1071-1088.

Sondik, E. J. (1978). "The optimal control of Partially Observable Markov Processes over the Infinite Horizon: Discounted Costs." *Operations Research* **26**(2): 282-304

Spaan, M. T. J. and N. Vlassis (2005). “Perseus: Randomized Point-based Value Iteration for POMDPs”. *Journal of Artificial Intelligence Research* 24 : 195-220.